Symbolic Variables

Table of Contents

In the previous section we learned how to create and use Matlab variables for storing numerical values during a Matlab session. Now we'd like to introduce a different kind of variable called a symbolic variable.

Declaring

These are necessary when the objects we are manipulating are not just numbers but rather expressions involving mathematical variables. For example, we would like to be able to use Matlab to manipulate a polynomial like `6*x^2+5*x-4`. The variable `x` that we see in this polynomial is not really a Matlab variable because it doesn't have a value. Rather it is a symbolic variable that is part of the polynomial we are going to be working with. In order for Matlab to manipulate mathematical expressions containing variables, we have to begin by creating these symbolic variables. For example, you could create the symbolic variable `x` like this:

```
syms x
```

You can create more than one symbolic variable at a time, like this:

```
syms x y z
```

Note that it is perfectly okay to declare a variable as symbolic more than once.

Now that we've created the symbolic variable `x`, Matlab will be able to manipulate mathematical expressions that contain `x`. Below we'll practice using symbolic variables in several useful ways.

Note: In all of the examples that follow we are going to be using the symbolic variable `x`. To get the examples to work, you must first create the symbolic variable as shown above.

Factoring Polynomials

Suppose you are trying to factor the polynomial `32*x^6 - 88*x^5 - 20*x^4 + 110*x^3 + 55*x^2 - 4*x - 4` using Matlab. No problem -- just use the `factor` function like this:

```
factor(32*x^6 - 88*x^5 - 20*x^4 + 110*x^3 + 55*x^2 - 4*x - 4)
```

```
ans =

(4*x - 1)*(x - 2)^2*(2*x + 1)^3
```

Recall that Matlab uses * for multiplication and ^ for exponentiation. Don't forget the * as Matlab will error if you forget it.

Using `pretty`

If you don't like seeing the * and the ^ in the output, you can pass the result of a command like `factor` into another command called `pretty`. Here's how that might look:

```
pretty(factor(32*x^6 - 88*x^5 - 20*x^4 + 110*x^3 + 55*x^2 - 4*x - 4))
```

```
                        2          3
    (4 x - 1) (x - 2)  (2 x + 1)
```

Expanding

The opposite of factoring is expanding. Suppose you have an expression with parentheses like `(7*k+2)*(5*k-3)*(k+7)` and you want to multiply it out. You can use the `expand` function as illustrated below. Note that we must first make sure `k` defined symbolically:

```
syms k
expand((7 * k + 2) * (5 * k - 3) * (k + 7))
```

```
ans =

35*k^3 + 234*k^2 - 83*k - 42
```

Simplifying

You can ask Matlab to try to simplify any mathematical expression. It doesn't always work! But many times you'll be impressed by how good Matlab is at this. Below are several examples illustrating the `simplify` function:

Try inventing other examples of your own!

```
simplify(3 * x + (x + 4)*(x - 7) + 15)
```

```
ans =

x^2 - 13
```

```
simplify((3 * x + 4) / (6 * x ^ 2 + 5 * x - 4))
```

```
ans =

1/(2*x - 1)
```

```
simplify(cos(x * 3)^2 + sin(x * 3)^2)
```

```
ans =

1
```

```
simplify(log(x) + log(2 * x))
```

```
ans =

log(2) + 2*log(x)
```

Do I Have to Declare Symbolic?

If you ask Matlab to factor a polynomial without first declaring the variables in it as symbolic you will get an error. This may seem strange but Matlab is just being sure. If you haven't told it what `x` is, how should it know what `x^2-4` is, for example?

Published with MATLAB® 8.0